UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Milan Derlink

Testiranje primernosti uporabe NoSQL v okviru računovodske aplikacije

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2014

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Milan Derlink

Testiranje primernosti uporabe NoSQL v okviru računovodske aplikacije

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Aljaž Zrnec

Ljubljana, 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Testiranje primernosti uporabe NoSQL rešitev v okviru računovodske aplikacije

Tematika naloge:

NoSQL podatkovne baze so v zadnjem času vedno bolj popularne, zaradi česar se v mnogih podjetjih brez premisleka odločajo za zamenjavo obstoječih relacijskih podatkovnih baz z novimi NoSQL rešitvami. V številnih situacijah je njihova učinkovitost sicer veliko višja kot pri relacijskih podatkovnih bazah, vendar to ne pomeni, da so te podatkovne baze primerne za reševanje kateregakoli problema. V diplomski nalogi primerjajte uporabo klasične - relacijske in NoSQL podatkovne baze v okviru aplikacije za računovodstvo. Osredotočite se predvsem na performančne lastnosti in podporo transakcijam. Ugotovite, ali je v takem tipu aplikacij, ki zahtevajo ACID transakcije, smiselno in sploh mogoče uporabiti podatkovno bazo MongoDB.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Milan Derlink, z vpisno številko **63090406**, sem avtor diplomskega dela z naslovom:

Testiranje primernosti uporabe NoSQL rešitev v okviru računovodske aplikacije

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Aljaža Zrneca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 15. septembra 2014 Podpis avtorja:

Rad bi se zahvalil vsem, ki ste mi pomagali na poti do moje diplome. Posebej bi se zahvalil mojemu mentorju, moji partnerici in Tomažu ter vsem na katere sem se lahko zanesel za pomoč v času študija.

Moji Manci, ki mi je stala ob strani tudi ob najtežjih trenutkih.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
NoSQL	Not Only SQL	ne-relacijske baze
SQL	Structured Query Language	Strukturirani poizvedovalni jezik
ACID	Atomicity, Consistency, Isolation, Durability	Atomarnost, Konsistentnost, Izolacija, Trajnost
SUPB	database management system	sistem za upravljanje podatkovnih baz
PHP	Hypertext Preprocessor	skriptni jezik
XML	EXtensible Markup Language	razširljiv označevalni jezik
BSON	Binary JSON	binarni JSON
JSON	JavaScript Object Notation	opis JavaScript objekta
HTTP	HyperText Transfer Protocol	protokol za prenos podatkov
HTML	Hyper Text Markup Language	jezik za označevanje nadbesedila

Povzetek

Relacijske baze obstajajo že od začetka sedemdesetih let prejšnjega stoletja. V tem času so se aplikacije, ki uporabljajo podatke shranjene v bazah, zelo spremenile. Postale so kompleksnejše, povečalo se je število sočasnih uporabnikov, povečala se je količina podatkov. V poznih devetdesetih letih se je pojavil internet. Z razmahom interneta, se je povečalo število sočasnih dostopov do podatkovnih baz in obseg podatkov, ki jih je potrebno hraniti in imeti na voljo, je zelo narasel. V diplomskem delu bomo predstavili relacijske baze, prednosti in slabosti. Predstavili bomo tudi baze NoSQL (Not Only SQL), ki so se pojavile kot odgovor na slabosti relacijskih baz in se pojavljajo v sodobnih internetnih aplikacijah. Kljub prednosti baz NoSQL pri uporabi v okviru sodobnih internetnih aplikacijah, bomo pokazali, da baze NoSQL niso primerne za vse sodobne internetne aplikacije. Ena izmed njih je internetna aplikacija za računovodski program. V diplomskem delu bomo pokazali, da so v tem primeru relacijske baze bolj primerne od NoSQL podatkovnih baz. Za testiranje primernosti NoSQL baze za uporabo v okviru računovodske aplikaciji bomo ustvarili enostavno aplikacijo, ki bo uporabljala podatke iz relacijske in NoSQL podatkovne baze. Merili bomo hitrost dostopa do podatkov z različnimi poizvedbami. Ne-relacijske baze so zaradi odsotnosti sheme bolj fleksibilne pri kasnejšem dodajanju atributov (novih vrst podatkov, ki jih v načrtovanju baze nismo predvideli) in omogočajo pa tudi enostavno razširjanje pomnilnega in diskovnega prostora z dodajanjem novih vozlišč v gručo. Relacijske baze so v tem pogledu bolj toge. Kljub temu, da veljajo ne-relacijske baze za hitrejše od relacijskih baz, bomo pokazali, da v primeru, ki ga predstavimo v diplomskem delu, to ne drži. V ta namen smo uporabili relacijsko podatkovno bazo MySQL in dokumentno NoSQL podatkovno bazo - MongoDB. Obe bazi vsebujeta enake podatke, in sicer podatke, ki so značilni za računovodsko področje. Za ugotavljanje učinkovitosti smo nad obema bazama izvajali iste transakcije. Vsaka od transakcij se je izvedla večkrat, na podlagi česar smo lahko izračunali povprečni čas njenega izvajanja. Računovodski programi so v osnovi namenjeni večinoma poizvedovanju. V diplomskem delu smo se osredotočili na poizvedbe, ki so del transakcij. Pripravili smo generator podatkov, ki smo jih vnesli v bazo. Pripravili smo tudi preprosto internetno računovodsko aplikacijo preko katere lahko vnašamo podatke. V računovodskem svetu se vneseni podatki ne smejo brisati oziroma popravljati. V tem duhu se torej nismo posvečali brisanju in popravljanju podatkov v bazi.

Ključne besede: SQL, NoSQL, MySQL, relacijska baza, dokumentna baza, računovodstvo, html, php, MongoDB, BSON, JSON.

Abstract

Relational databases have existed since the beginning of the seventies of the last century. In the meantime, applications that use data stored in databases have changed. They have become more complex, an increasing number of concurrent users, increase the amount of data. In the late nineties, the internet emerged. With the expansion of the Internet, the number of concurrent accesses to databases and the volume of data that need to be kept and be made available have increased. In the thesis we present a relational database, strengths and weaknesses. We will present the database NoSQL which emerged as a response to the weaknesses of relational databases in modern Internet applications. Despite the advantages of NoSQL databases when used in the context of modern Internet applications, we show that NoSQL databases are not suitable for all modern Internet applications. One of them is an Internet application for the accounting program. In the thesis we show that in this case the relational database is more suitable than NoSQL databases. To test the suitability of NoSQL databases for use within the accounting application, we will create a simple application that will use data from relational and NoSQL databases. We shall measure the speed of access to data with different queries. Non-relational databases due to the absence of the scheme are more flexible for subsequent addition of attributes (new types of data that in the planning of database we did not predict) and allow easy dissemination of storage and disk space by adding new nodes in the cluster. Relational databases are in this respect more rigid. Despite that non-relational database should be faster than relational databases, we show that in the case presented by the thesis, this is not true. To this end, we used a relational database MySQL and NoSQL document database - MongoDB. Both databases contain the same information, namely information that is specific for accounting. To determine the effectiveness of the two bases, we performed the same transactions. All transactions are carried out repeatedly, on the basis of which we can calculate the average time of its implementation. Accounting programs are basically designed mainly for searching data from database. In the thesis we focus on the queries that are part of the transaction. We have prepared a data generator for data that we have entered into the database. We have also prepared a simple internet accounting application through which we can enter data. In the financial world, the entered data may not be deleted or repaired. In this spirit, therefore we did not delete and update data in the database.

Keywords: SQL, NoSQL, MySQL, relational database, document database, accounting, html, php, MongoDB, BSON, JSON.

1. Uvod

Podatkovne baze tipa NoSQL so v zadnjem času zelo popularne. Zaradi svoje popularnosti se v marsikaterem podjetju prehitro in nepremišljeno odločajo za zamenjavo relacijskih baz z bazami NoSQL. Zamenjujejo jih tudi tam, kjer ne bi bilo potrebno oziroma še slabše, zamenjujejo jih tam, kjer se obnesejo slabše kot, že obstoječe relacijske baze.

Cilj diplomskega dela je pokazati, da je uporaba relacijskih baz v določenih aplikacijah primernejša kot pa sodobnejše baze NoSQL.

Diplomsko delo je razdeljeno na šest poglavij. V drugem poglavju bomo predstavili NoSQL podatkovno bazo MongoDB in relacijsko podatkovno bazo MySQL. Navedli bomo prednosti in slabosti obeh tipov podatkovnih baz.

V tretjem poglavju bomo predstavili razvoj podatkovne baze MySQL. Opisali bomo razvoj NoSQL podatkovne baze MongoDB. Pokazali bomo tudi, da je relativno lahko prenesti podatke iz podatkovne baze MySQL v MongoDB.

V četrtem poglavju bomo predstavili transakcije s katerimi bomo testirali hitrost obeh baz. Razvili bomo preprosto internetno aplikacijo s pomočjo katere bomo testirali hitrosti obeh baz. Predstavili bomo tudi rezultate testiranja

V petem poglavju bomo predstavili in obrazložili rezultate testiranja.

V zaključku bomo podali sklepne ugotovitve.

2. Podatkovne baze

2.1 Splošno o podatkovnih bazah

2.1.1 Definicija

Obstaja več definicij podatkovnih baz. Omenili bomo tri.

- Podatkovna baza je zbirka povezanih podatkov. Podatki so dejstva, so shranjena na nekem računalniškem trajnem pomnilniku, ki se jim lahko pripiše pomen (ki implicitno imajo pomen). (Elmasri and Navathe)[5].

- Podatkovna baza je upravljana zbirka povezanih podatkov, shranjena na računalniškem sistemu, deljena med več uporabniki, zaščitena z varnostnimi mehanizmi in shranjena z nadzorovano redundantnostjo. (Stamper and Price [5].

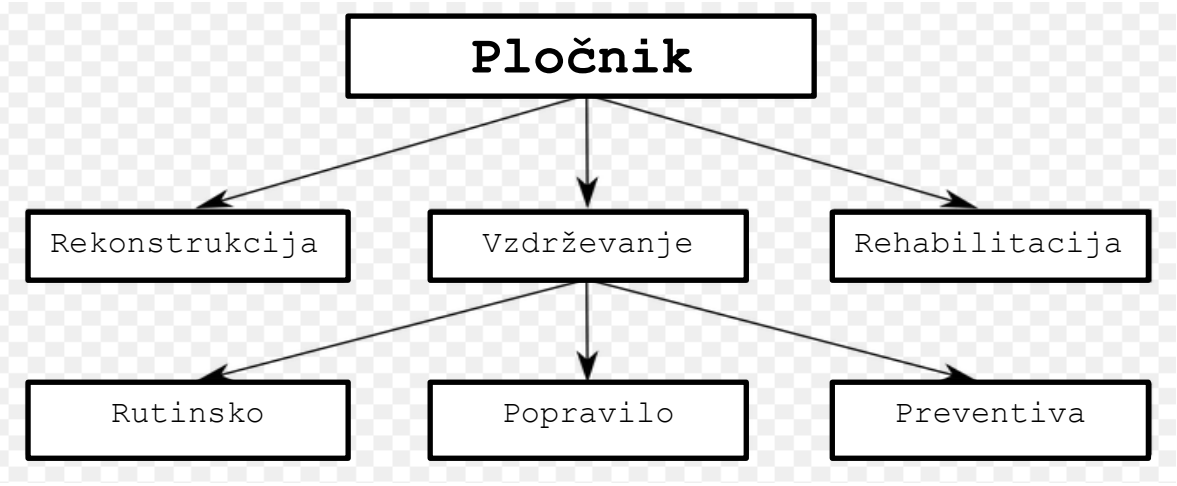
- Podatkovna baza je organizirana zbirka logično povezanih podatkov in opisov le teh, načrtovana tako, da zadovoljuje informacijske potrebe organizacije. (Connolly and Begg)[5].

2.1.2 Zgodovina podatkovnih baz

V šestdesetih letih prejšnjega stoletja Charles Bachman predstavi prvi mrežni podatkovni model. V istem desetletju IBM razvije hierarhični podatkovni model.

Hierarhični model podatke organizira v drevesni strukturi. Ti podatki se lahko povezujejo z razmerji tipa 1:N, kjer ima en . Imamo sistem starš, ki ima lahko več potomcev. Podatki so shranjeni v zapisih, ki so med seboj povezani. Vsak zapis v hierarhičnem modelu je ekvivalenten vrstici v relacijski bazi, tip podatka, pa stolpcu v relacijski bazi. IBM (International Business Machines) predstavi IMS (Information Management System), ki je uporabljal ta podatkovni model.

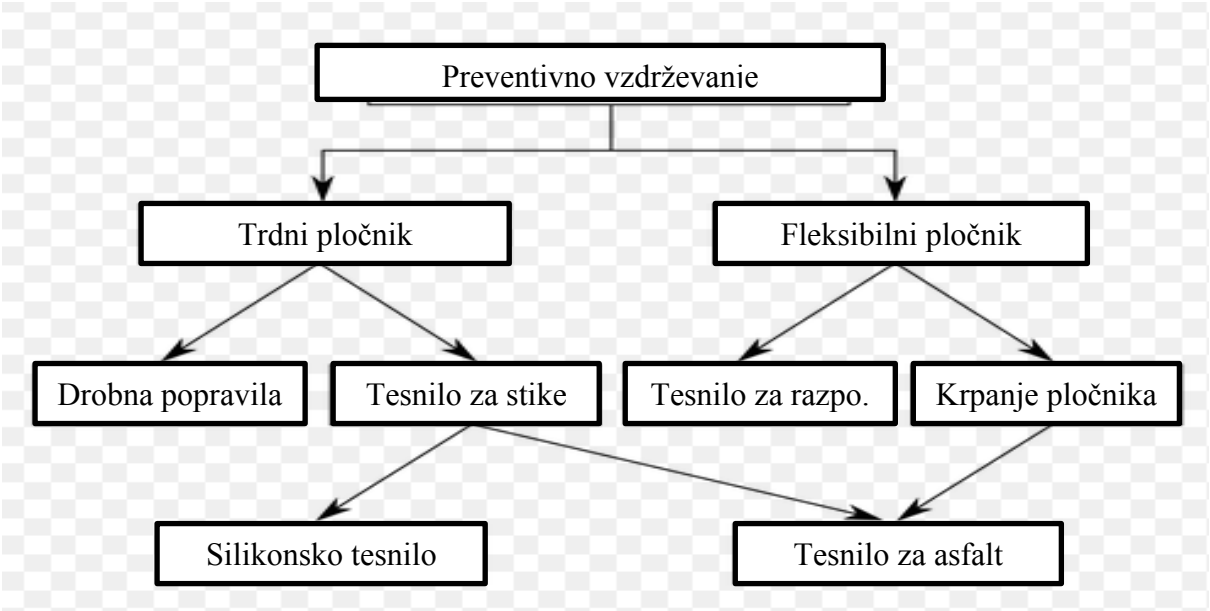
Slika 2.1 prikazuje primer hierarhičnega podatkovnega modela.



Slika 2.1: Primer hierarhičnega modela[6].

Mrežni model, ki je nastal istočasno kot hierarhični model, je omogočal podatke povezovati z uporabo razmerij tipa mnogo proti mnogo M:N. Do podatka v mrežnem modelu dostopamo neposredno, poznati moramo ključe oziroma kazalce. S kazalcem izračunavamo lokacije nekega drugega podatka.

Slika 2.2 prikazuje primer mrežnega podatkovnega modela.



Slika 2.2: Primer mrežnega modela[11].

Relacijski podatkovni model, ki se pojavi leta 1970, bomo podrobneje predstavili v naslednjem poglavju.

V osemdesetih letih prejšnjega stoletja se razvije tudi poizvedovalni jezik SQL. Uporablja se za poizvedovanje v relacijskih bazah.

Z razmahom internetnih uporabnikov na začetku 21. stoletja, pa so relacijske baze, ki niso bile namenjene servisiranju tako velikega števila uporabnikov, počasi postajale vedno težje uporabne. Kot odgovor na težave, ki so jih imele relacijske podatkovne baze s porazdeljeno zasnovo Interneta, se pojavijo se tako imenovane NoSQL podatkovne baze. Te so sposobne servisirati ogromno uporabnikov istočasno. Trenutno največji svetovni ponudniki internetnih storitev uporabljajo za svoje delovanje eno ali več izmed NoSQL rešitev.

2.2 Relacijska baza

Relacijski podatkovni model se pojavi leta 1970. Model predstavlja množico med seboj povezanih relacij. Vsaka relacija predstavlja množico resničnih trditev. Lastnosti relacijskega podatkovnega modela so naslednje:

- Vrednosti v relacijah so atomarne[3].
- Vsaka vrstica je unikatna[3].
- Vrednosti v stolpcu so istega tipa[3].
- Vrstni red stolpca ni pomemben[3].
- Vrstni red v vrstici ni pomemben[3].
- Vsak stolpec ima unikatno ime[3].

Za zanesljivost pri transakcijah v relacijskih bazah skrbi tako imenovani ACID (Atomicity, Consistency, Isolation, Durability).

- Atomarnost (Atomicity)

  Vsaka transakcija je sestavljena iz več operacij. Če se vse operacije izvedejo uspešno, je transakcija uspela. SUPB zagotovi atomarnost. Kadar transakcija ni uspela, vse

operacije se ne izvedejo, SUPB (sistem za upravljanje podatkovnih baz) razveljavi transakcijo. Vrne se v prvotno stanje (Rollback).

- Konsistentnost (Consistency)

  Podatkovna baza pri izvajanju transakcije prehaja iz enega stanja v drugega. Pri kršitvah se vse spremembe razveljavijo.

- Izolacija (Isolation)

  Ker bi lahko transakcije med izvajanjem vplivale na druge transakcije, ne smejo biti vidne. Šele ko so potrjene postanejo vidne.

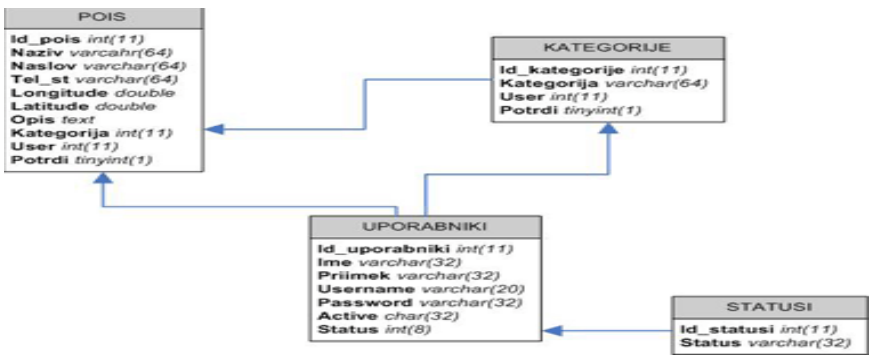
- Trajnost (Durability)

  Da bi preprečili izgubo podatkov pri odpovedi sistema, se spremembe zapišejo na disk preden je transakcija potrjena.

Relacijska podatkovna baza je sestavljena iz relacij. Relacijo predstavlja enolična tabela. V tabeli so atributi oziroma stolpci in zapisi oziroma vrstice. Vsakemu stolpcu je določen podatkovni tip. S primarnim ključem določimo vsako vrstico v tabeli. Tabele se med sabo povezujejo s tujimi ključi. Podrejena tabela vsebuje tuji ključ, ki je primarni ključ v nadrejeni tabeli.

Normalizacija je proces s katerim odstranimo podvojitev podatkov. Če se podatki podvajajo stolpce premestimo v nove tabele. Bolj ko so podatki normalizirani počasnejša je baza. Denormalizacija pohitri bazo. Slaba stran je, da obstajajo podvojeni podatki.

Slika 2.3 prikazuje relacijski podatkovni model.



Slika 2.3: Relacijski podatkovni model[17].

2.2.1 Poizvedovalni jezik SQL

SQL ali strukturirani povpraševalni jezik za delo s podatkovnimi bazami je najbolj razširjen in standardiziran povpraševalni jezik za delo s podatkovnimi zbirkami, s programskimi stavki, ki posnemajo ukaze v naravnem jeziku. Določen je z ANSI/ISO SQL standardom. SQL standard se je razvijal od leta 1986 in danes obstaja več različic. Oznaka SQL-92 se navezuje na standard izdan v letu 1992, SQL:1999 se navezuje na standard izdan leta 1999, SQL:2003 se navezuje na različico iz leta 2003 in tako naprej. Izraz SQL standard uporabljamo za poimenovanje trenutne različice SQL standarda v vsakem časovnem obdobju[19].

Najbolj pogosta operacija v SQL-u je poizvedba, ki se izvrši s SELECT stavkom. SELECT stavek vrne podatke iz ene ali več tabel. Standardni SELECT stavek nima nobenega vpliva na podatke v podatkovni bazi in jih ne spreminja. Poizvedba omogoča uporabniku, da opiše strukturo želenih podatkov. SUPB je odgovoren za planiranje, optimiziranje in izvajanje fizičnih operacij, potrebnih za izpis želenega rezultata. Optimizacija SQL stavkov lahko znatno pohitri izvajanje poizvedb[19].

Struktura SELECT stavka:

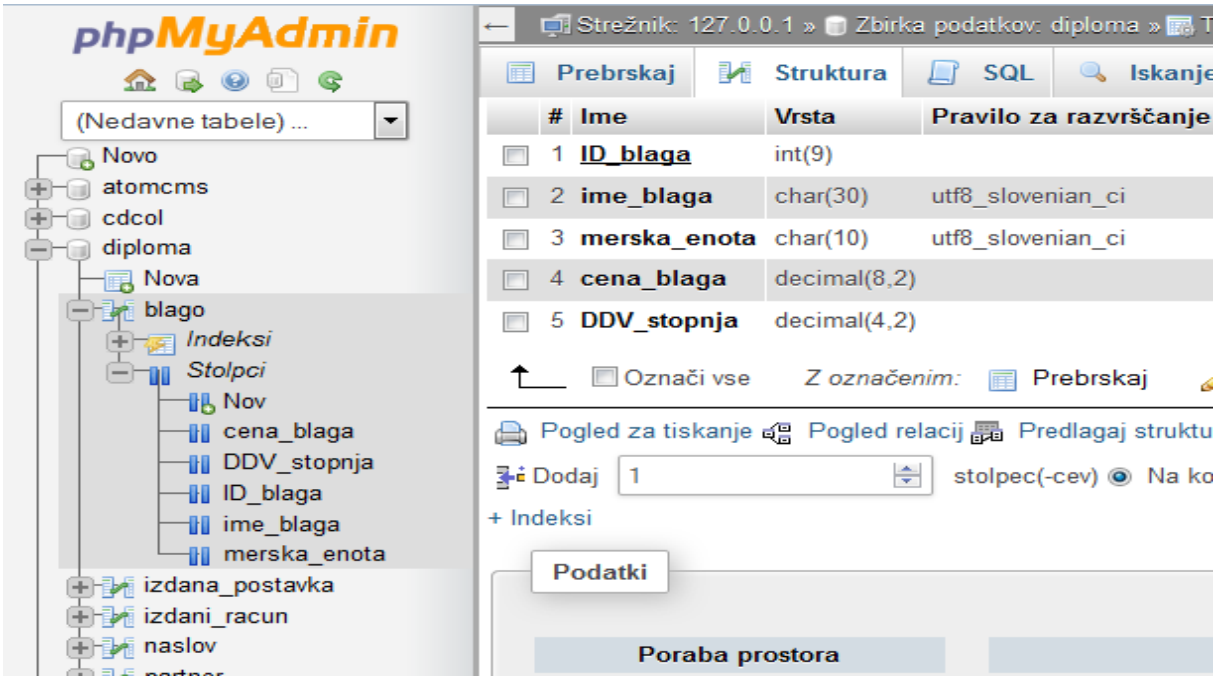
- SELECT stavku lahko sledi znak *, ki pomeni, da se bodo izpisala vsa polja iz tabele oziroma tabel v FROM stavku. Namesto zvezdice lahko napišemo imena polj,
- FROM stavku sledijo imena tabel, iz katerih bomo vzeli podatke,
- WHERE stavek vsebuje pogoje, ki jih bo upoštevala poizvedba,
- GROUP BY projicira vrstice z istimi vrednostmi v manjšo množico vrstic,
- HAVING stavek filtrira vrstice,
- ORDER BY stavek sortira rezultat poizvedbe. Stavku ORDER BY lahko sledi eno ali več polj. Sortiranje je lahko naraščajoče (ASC) ali padajoče (DESC)[1].

Zaradi svojih značilnosti in enostavnega poizvedovalnega jezika SQL, postanejo relacijske podatkovne baze vse bolj razširjene.

2.2.2 Relacijska baza MySQL

Ena izmed relacijskih baz je tudi odprtokodna relacijska baza MySQL. To relacijsko podatkovno bazo bomo uporabili za naše namene testiranja. Baza MySQL streže podatke s strežniške strani. Za transakcije uporabljamo SQL. Ima zelo dobro podporo PHP (Hypertext Preprocessor). Podatki v bazi so strukturirani. Podatkovno bazo je mogoče upravljati preko konzole z uporabo ukazov SQL ali pa za ta namen uporabimo orodje phpMyAdmin, ki predstavlja spletni odjemalec za to podatkovno bazo.

Slika 2.4 prikazuje grafični vmesnik phpMyAdmin.



Slika 2.4: Grafični vmesnik phpMyAdmin.

2.3 NoSQL baza podatkov

NoSQL baze podatkov so uporabne v porazdeljenem okolju. V primerjavi z relacijskimi bazami imajo manjši nabor funkcij. Tudi ACID princip ni popolnoma podprt. V primerjavi z relacijskimi bazami nimamo stikov, bolj kompleksnih filtrov in agregacije. Vse to moramo implementirati v aplikaciji, ki smo jo ustvarili za testiranje naše baze.

2.3.1 Dokumentni model

Dokumentno orientirana baza podatkov je računalniški program namenjena za shranjevanje, vračanje in upravljanje dokumentno orientiranih podatkov. Dokumentne baze so v zadnjem

času zelo popularne. Od relacijskih baz se v največji meri razlikujejo v tem, da podatki nimajo v naprej opredeljene podatkovne sheme. Podatkovno enoto v podatkovno orientirani podatkovni bazi predstavlja dokument[4]. Dokument lahko vsebuje veliko različnih podatkov. En dokument v dokumentno orientirani bazi je primerljiv z vrstico v relacijski bazi. Razlika je, da je tip podatkov v relacijski bazi določen že na samem začetku pri razvoju baze. Lahko se zgodi, da kasneje, ko že dalj časa uporabljamo relacijsko bazo, pride potreba, da želimo shranjevati še kak atribut, ki ga v razvoju relacijske baze nismo upoštevali in implementirali. Naknadno dodajanje atributov je zahtevno in zamudno. Teh problemov v dokumentno orientirani bazi nimamo. Dokument nima omejitve koliko in kakšne vrste podatkov vsebuje dokument. Standardi, ki se uporabljajo so XML (EXtensible Markup Language), BSON (Binary JSON), JSON (JavaScript Object Notation).

2.3.2 MongoDB

MongoDB je odprtokodna podatkovna baza NoSQL. Baza je dokumentnega tipa. Napisana je v programskem jeziku C++. Razširitev navzven je sorazmeroma enostavna, omogoča pa tudi funkcionalnosti, ki so značilne za relacijske baze. Sortiranje, sekundarno indeksiranje[8]. Bazo lahko upravljamo preko konzole mongo shell, lahko pa uporabimo programe z grafičnim vmesnikom. V nadaljevanju so navedene nekatere značilnosti:

- Dokumentni model

  Podatki so shranjeni v dokumentih tipa BSON (Binary JSON). V relacijskih bazah je to ekvivalent vrstici v tabeli. Dokument je shranjen v zbirki. Vsak dokument vsebuje polja, kar predstavlja en podatek. Polja lahko gnezdimo znotraj dokumenta. Dokument predstavlja množico dvojic »ime atributa : vrednost«[2].

- Agregacija

  MongoDB ponuja več agregacijskih orodij. Count, ki prešteje število dokumentov v zbirki. Distinct poišče določene vrednosti, ki smo jih določili s ključem. MapReduce omogoča iskanje dokumentov na več različnih strežnikih. Slaba stran tega orodja je, da je počasno in naj se ga ne bi uporabljalo za iskanje v realnem času[2].

- Regularni izrazi

  So zelo uporabni pri primerjanju nizov.

- Indeksiranje

Z indeksiranjem pohitrimo iskanje podatkov v dokumentih.

- Replikacija

  MongoDB uporablja replikacijo tipa gospodar – suženj. To je zelo uporabno pri varnostnih kopijah, izpadih, povečanem dostopu za branje podatkov[2].

- Porazdeljevanje obremenitve.

  Pri zelo velikih bazah je priporočljivo porazdeliti podatke na več strežnikov.

V tabeli 2.1 bomo predstavili terminologijo baze MongoDB in njene ekvivalente v bazi MySQL.

MongoDB terminologija	**MySQL terminologija**
baza podatkov	*baza podatkov*
zbirka	*tabela*
dokument ali BSON dokument	*vrstica*
polje	*stolpec*
indeks	*indeks*
gnezdeni dokumenti	*stiki tabel*
primarni ključ	*primarni ključ*

Tabela 2.1: Terminologija baze MongoDB in baze MySQL.

V tabeli 2.2 bomo predstavili nekaj ukazov, ki so značilni za SQL in kako se uporabljajo v lupini MongoDB.

MongoDB terminologija	**MySQL terminologija**
INSERT INTO USERS VALUES(3,5)	*db.users.insert({a:3,b:5})*
SELECT a,b FROM users	*db.users.find({}, {a:1,b:1})*
*SELECT * FROM users*	*db.users.find()*
*SELECT * FROM users WHERE age>33 AND age<=40*	*db.users.find({'age':{$gt:33,$lte:40}})*

SELECT DISTINCT last_name FROM users	*db.users.distinct('last_name')*
*SELECT COUNT(*y)* *FROM users*	*db.users.count()*
CREATE INDEX myindexname ON users(name)	*db.users.ensureIndex({name:1})*

Tabela 2.2: Primerjava SQL in MongoDB ukazov.

3. Razvoj relacijske in dokumentne baze

3.1 Spletni strežnik in relacijska baza

Za namene testiranja baz smo ustvarili preprosto spletno stran, preko katere se poizvedbe pošiljajo na podatkovne strežnike. Uporabili smo odprtokodno distribucijo strežnika Apache XAMPP. Distribucija vsebuje Apache HTTP (HyperText Transfer Protocol) strežnik, relacijsko bazo MySQL, PHP in Pearl. Je zelo enostavna za namestitev, namestimo jo lahko na različne operacijske sisteme. Po namestitvi imamo takoj na razpolago http strežnik, relacijsko bazo in PHP.

Apache je spletni strežnik, ki igra ključno vlogo pri širjenju spleta. Bil je prva alternativa Netscapeovemu spletnemu strežniku, trenutno znanemu kot spletni strežnik Sun Java System. Od aprila 1996 je Apache najbolj popularen HTTP strežnik na celem spletu. Od oktobra 2007 pa je bilo na Apachijevih strežnikih postavljenih približno 48 % vseh spletnih strani[18].

Sam Apache strežnik smo namestili na lokalni računalnik. V našem primeru smo se odločili, da bomo testirali baze preko preproste internetne strani. Ker je strežnik na lokalnem računalniku, bosta tako odjemalec – klient in strežnik na istem računalniku.

PHP je skriptni jezik za razvoj internetnih strani, ki deluje na strežniku. Uporablja se tudi kot splošni programski jezik. Ustvaril ga je leta 1994 Rasmus Lerdorf. Ime PHP je na začetku pomenil Personal Home Page, danes pa pomeni Hypertext Preprocessor. PHP koda se lahko integrira v HTML (Hyper Text Markup Language) kodo.

MySQL je sistem za upravljanje s podatkovnimi bazami. MySQL je odprtokodna implementacija relacijske podatkovne baze, ki za delo s podatki uporablja jezik SQL. MySQL deluje na principu odjemalec - strežnik, pri čemer lahko strežnik namestimo kot sistem, porazdeljen na več strežnikov. Obstaja veliko število odjemalcev, zbirk ukazov in programskih vmesnikov za dostop do podatkovne baze MySQL[10].

3.2 Razvoj baze MySQL

Za potrebe testiranja smo razvili relacijsko bazo MySQL, ki vsebuje podatke o prejetih računih, ki jih podjetje prejema od svojih poslovnih partnerjev. Relacijska baza MySQL vsebuje tudi podatke o izdanih računih, ki jih podjetje izdaja svojim poslovnim partnerjem.

Podatki na prejetih računih so:

- podatki o izdajatelju računa (naziv, naslov, davčna številka),
- podatki o prejetem računu (številka računa, datum prejetega računa, rok plačila, datum opravljene storitve),
- podatki o blagu oziroma storitve, ki so na računu (ime blaga, merska enota, količina, cena, DDV stopnja).

Podatki na izdanih računih, ki jih izda podjetje pa so:

- podatki o izdajatelju računa (naziv, naslov, davčna številka),
- podatki o izdanem računu (številka računa, datum izdanega računa, rok plačila, datum opravljene storitve),
- podatki o blagu oziroma storitve, ki so na računu (ime blaga, merska enota, količina, cena, DDV stopnja).

Relacijsko bazo MySQL smo kreirali s pomočjo orodja za načrtovanje in ustvarjanje relacijskih baz Sybase PowerDesigner[20]. Bazo smo načrtovali in kreirali v dveh korakih, konceptualno in fizično načrtovanje.

3.2.1 Konceptualno načrtovanje

Najprej smo identificirali entitetne tipe ki smo jih potrebovali. V našem primeru so to:

- partner,
- naslov,
- posta,
- izdani_racun,

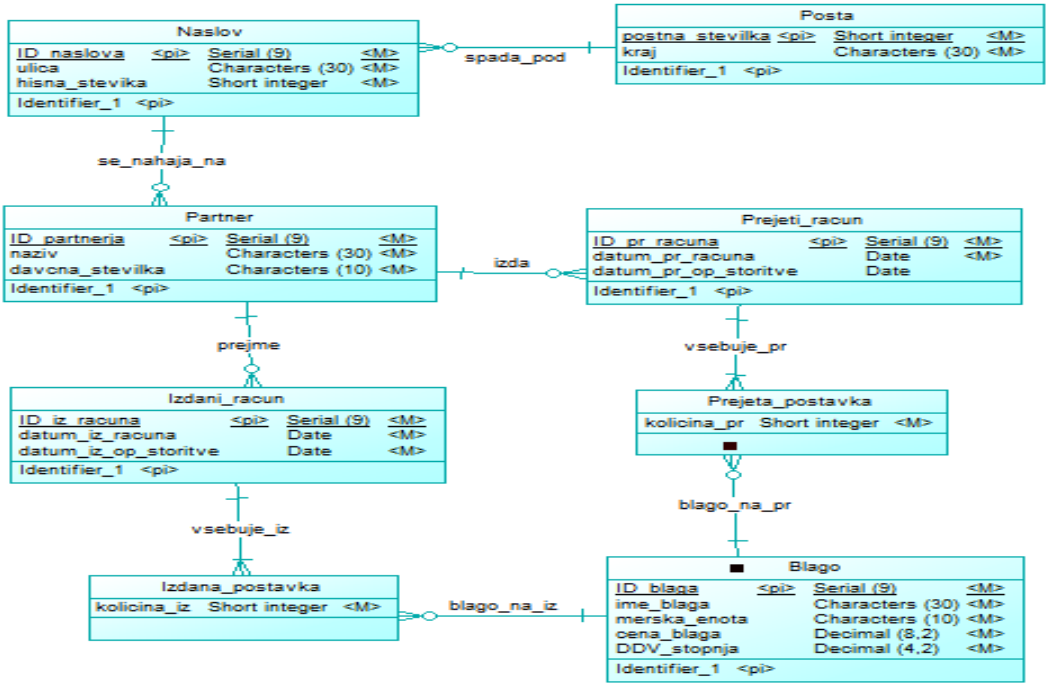
- prejeti_racun,
- izdana_postavka,
- prejeta_postavka,
- blago.

Izdana_postavka in prejeta postavka sta vmesni entiteti. Vpeljali smo jih, ker sta povezavi med entitetnim tipom izdani_racun in blago ter prejeti_racun in blago, povezave mnogo proti mnogo.

Ko smo identificirali entitetne tipe, smo identificirali povezave med entitetnimi tipi. V našem primeru je osem povezav med tipi:

- več partnerjev se nahaja na enem naslovu, oziroma na enem naslovu se nahaja več partnerjev,
- več naslovov se nahaja na eni pošti, oziroma ena pošta ima več naslovov,
- en partner prejme več naših izdanih računov, oziroma več naših izdanih računov prejme en partner,
- en izdani račun vsebuje več izdanih postavk, oziroma več izdanih postavk vsebuje en izdani račun,
- eno blago je lahko na več izdanih postavkah, oziroma na več izdanih postavkah je eno blago,
- en partner izda več naših prejetih računov, oziroma več naših prejetih računov izda en partner,
- en prejeti račun vsebuje več prejetih postavk, oziroma več prejetih postavk vsebuje en prejeti račun,
- eno blago je lahko na večih prejetih postavkah, oziroma na večih prejetih postavkah je eno blago.

Slika 3.1 prikazuje konceptualni podatkovni model.

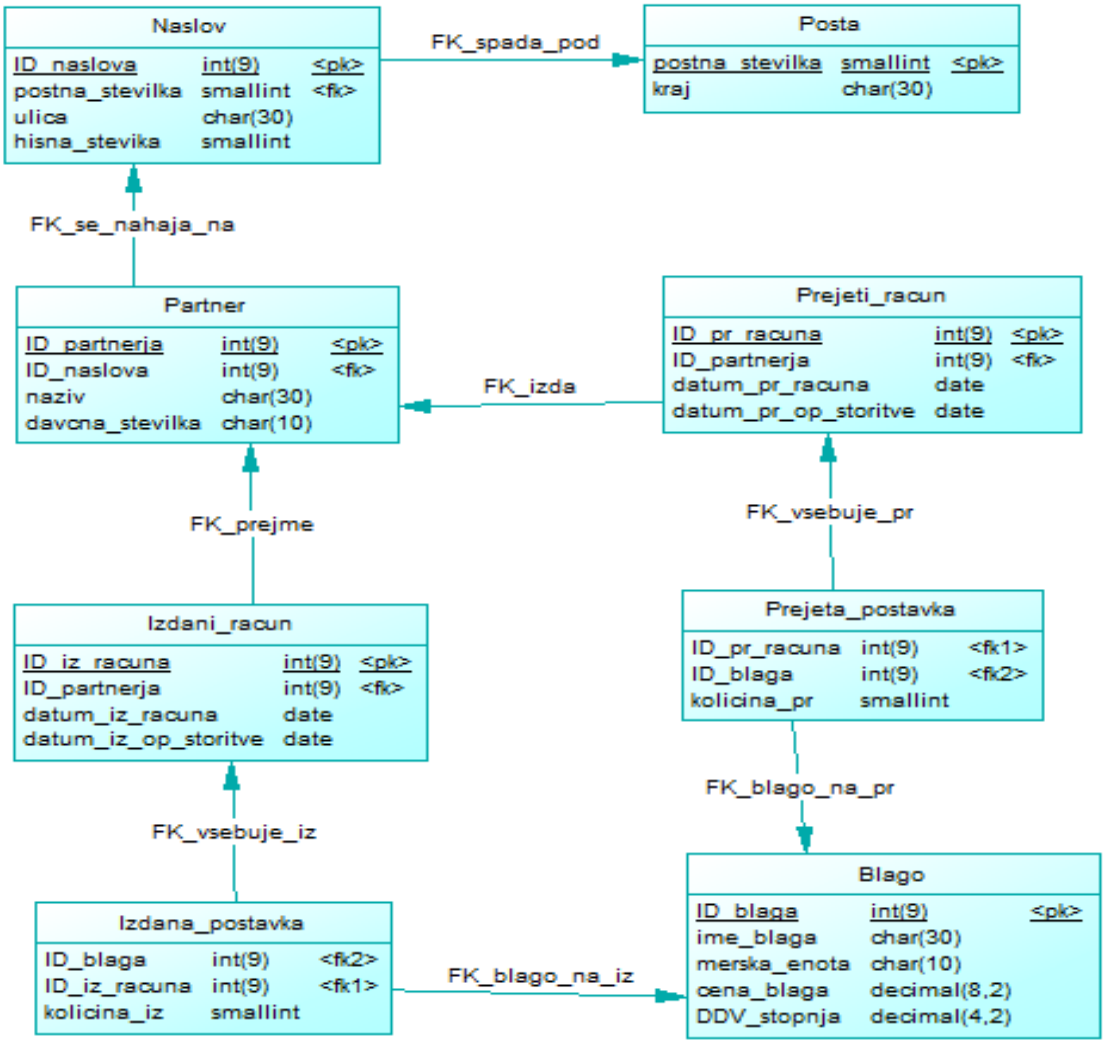


Slika 3.1: Model konceptualnega načrtovanja.

3.2.2 Fizično načrtovanje

Pri fizičnem načrtovanju smo podatkovni model, ki smo ga izdelali v okviru konceptualnega načrtovanja, pretvorili v fizični model, ki je namenjen točno določenemu SUPB. V našem primeru za MySQL. Za pretvorbo smo uporabili orodje Sybase PowerDesigner. S tem orodjem smo določili na kakšen način bodo v ciljnem SUPB, osnovne relacije predstavljene. Za vsako relacijo se opredelijo naziv in seznam osnovnih atributov, primarni ključ, tuji ključ in omejitve povezav.

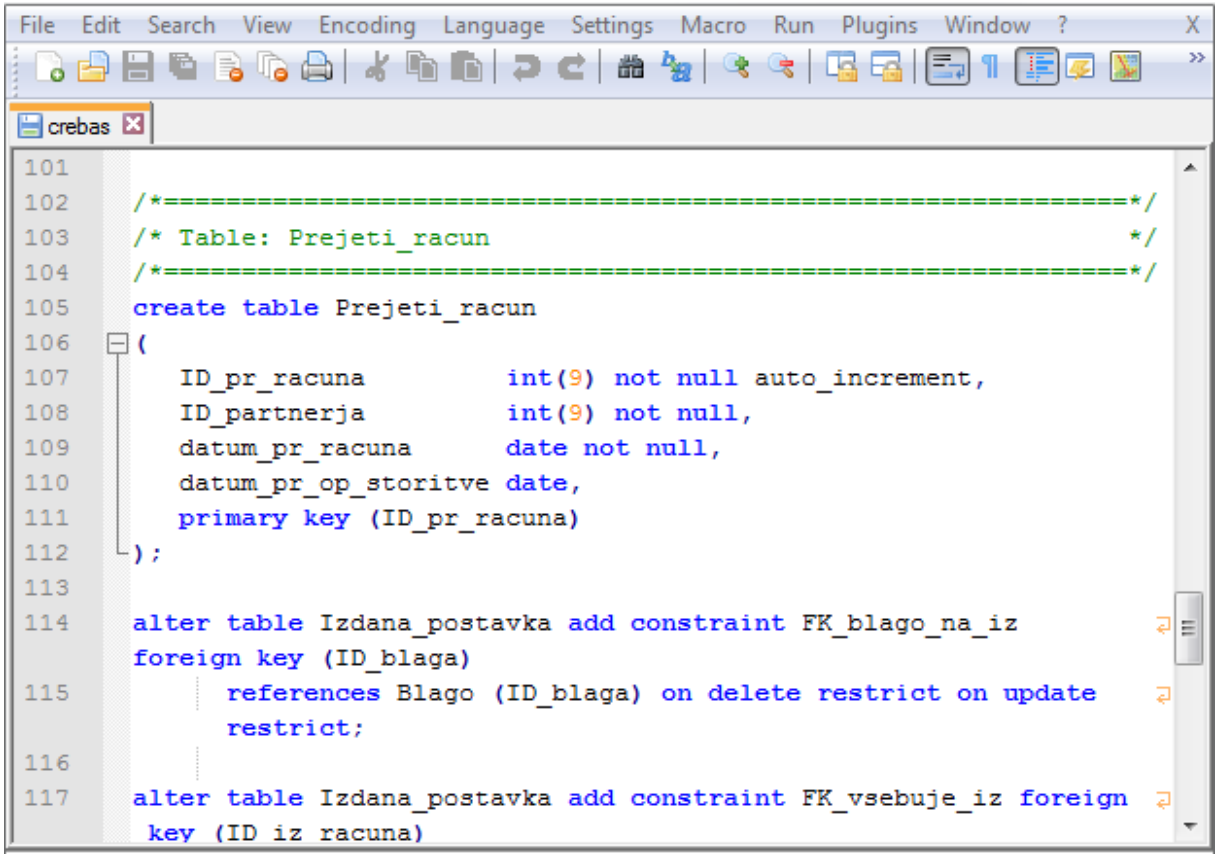
Slika 3.2 prikazuje fizični podatkovni model:



Slika 3.2: Model fizičnega načrtovanja.

S pomočjo orodja za načrtovanje baz Sybase PowerDesigner generiramo SQL skripto, ki smo jo uporabili za ustvarjanje baze na MySQL strežniku.

Slika 3.3 prikazuje del SQL skripte, ki ustvari bazo na MySQL strežniku.



Slika 3.3: SQL skripta.

3.3 Razvoj podatkovne baze MongoDB

MongoDB smo namestili na lokalni računalniku (localhost). Usposobitev dokumentne baze MongoDB je relativno enostavna. V sami fazi testiranja smo najprej ustvarili relacijsko bazo MySQL. To bazo smo napolnili s testnimi podatki. Da bi bili obe bazi čim bolj primerljivi med seboj, smo podatke iz relacijske baze uvozili v MongoDB. V bazi MySQL imamo možnost, da lahko podatke izvažamo v JSON formatu.

JSON je odprti standard, ki vsebuje človeku prijazen tekst za branje. Vsebuje objekte, ki so sestavljeni iz parov atribut – vrednost[7].

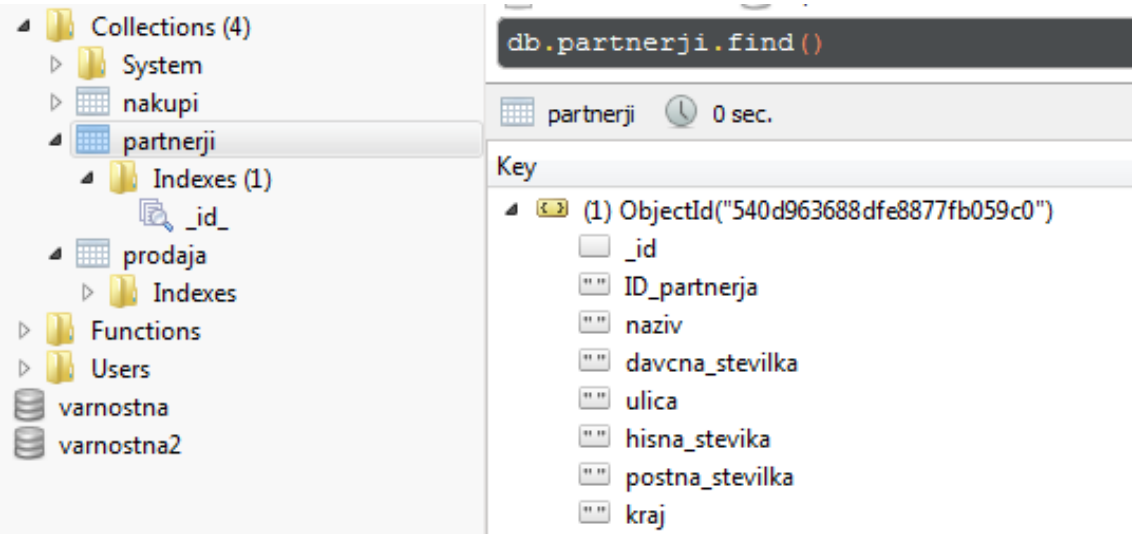
V dokumentni bazi MongoDB smo ustvarili tri zbirke (collection), vsaka zbirka pa ima različno število polj (field) v katerih so shranjeni podatki:

- partner, v zbirki je osem polj (_id, ID_partnerja,, naziv, davcna_stevilka, ulica, hisna_stevilka, postna_stevilka, kraj),

- nakupi, v zbirki je enajst polj (_id, cena_blaga, DDV_stopnja, ID_blaga, ime_blaga, merska_enota, kolicina_pr, datum_pr_op_storitve, datum_pr_racuna, ID_pr_racuna, ID_partnerja),

- prodaja, v zbirki je enajst polj ((_id, cena_blaga, DDV_stopnja, ID_blaga, ime_blaga, merska_enota, kolicina_iz, datum_iz_op_storitve, datum_iz_racuna, ID_iz_racuna, ID_partnerja).

Slika 3.4 prikazuje grafični vmesnik za nadzor podatkovne bazeMongoDB.



Slika 3.4: Grafični prikaz baze MongoDB.

Podatke smo prenesli iz MySQL baze s pomočjo JSON datoteke.

Uporabili smo obliko JSON, ker ima dokumentna baza MongoDB možnost uvoza v tej obliki. Ker so bile JSON datoteke večje od 16MB smo datoteke razdelil na več manjših datotek. To je bilo potrebno zaradi omejitve dokumentne baze MongoDB, ki ne dovoli uvoza datotek, ki so večje od 16MB. Baza vsebuje program mongoimport.exe s pomočjo katerega uvozimo podatke. Ukaz je sestavljen iz imena programa, imena zbirke (collection), ki naj bo ustvarjena in poti kjer se nahajajo JSON datoteka s podatki za uvoz.

4. Performančna analiza

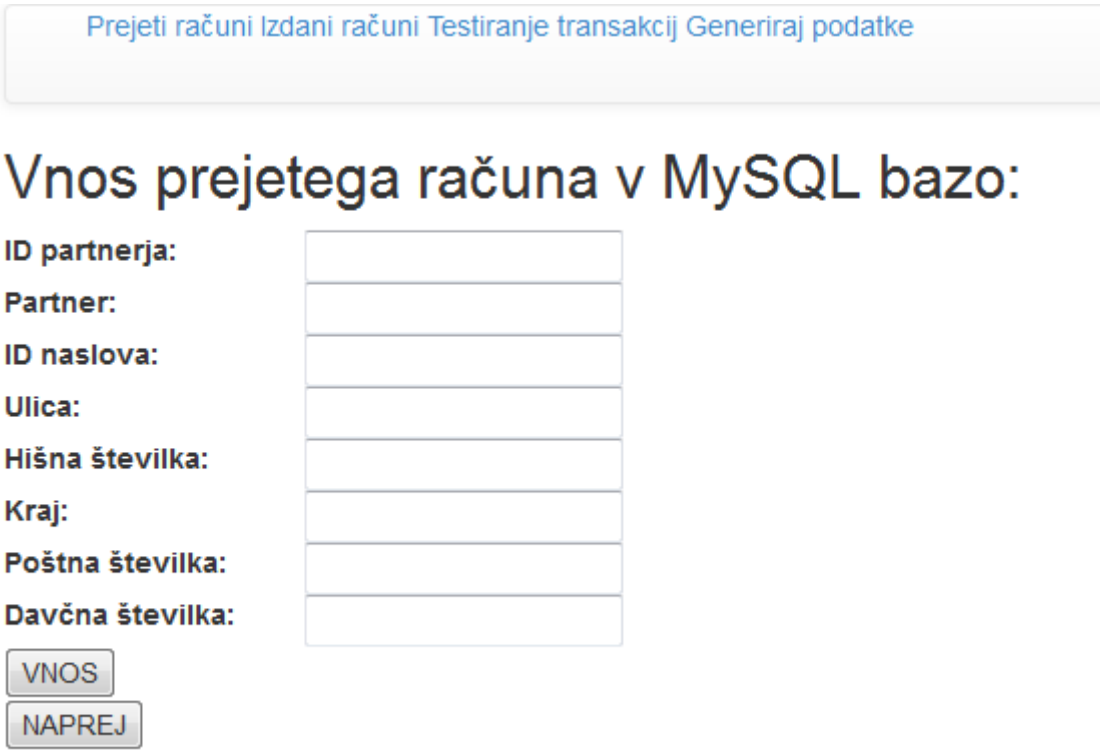
4.1 Preprosta internetna aplikacija za testiranje

Aplikacija je sestavljena iz treh delov. Prvi del se uporablja za generiranje podatkov, ki se nato zapišejo v bazo MySQL . Imamo možnost polnjenja vsake tabele posebej in nam tudi kaže koliko podatkov je v vsaki tabeli. Drugi del se uporablja za shranjevanje podatkov v bazi MySQL in MongoDB. Tretji del pa pošlje poizvedbe bazi MySQL in MongoDB, ter izpiše podatke koliko časa v milisekundah sta potrebovali obe bazi za vsako poizvedbo.

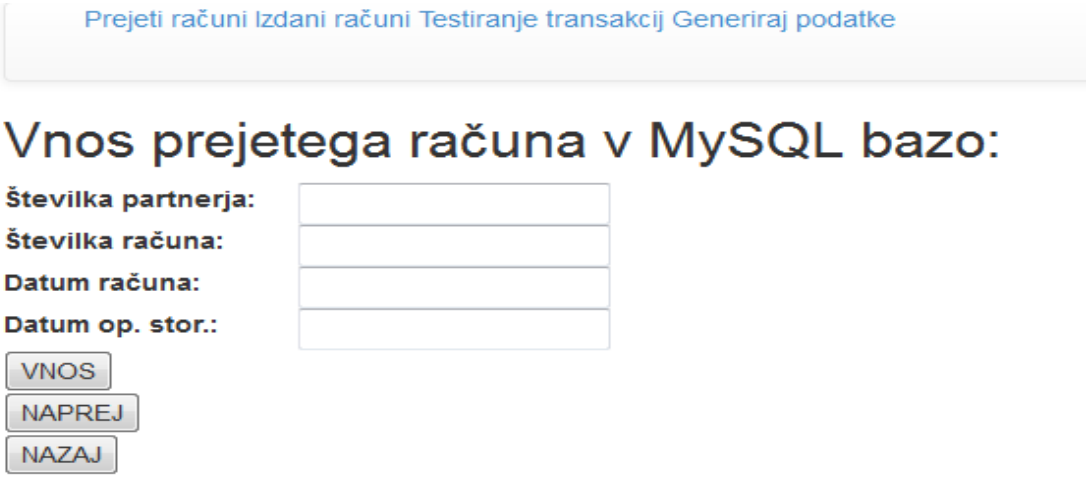
Podatki se generirajo s pomočjo PHP skripte, ki naključno pobira podatke iz naprej določenih seznamov. S SQL stavkom INSERT INTO vnesemo podatke v MySQL bazo. Slika 4.1 prikazuje ta del aplikacije.

Slika 4.1: Generiranje podatkov.

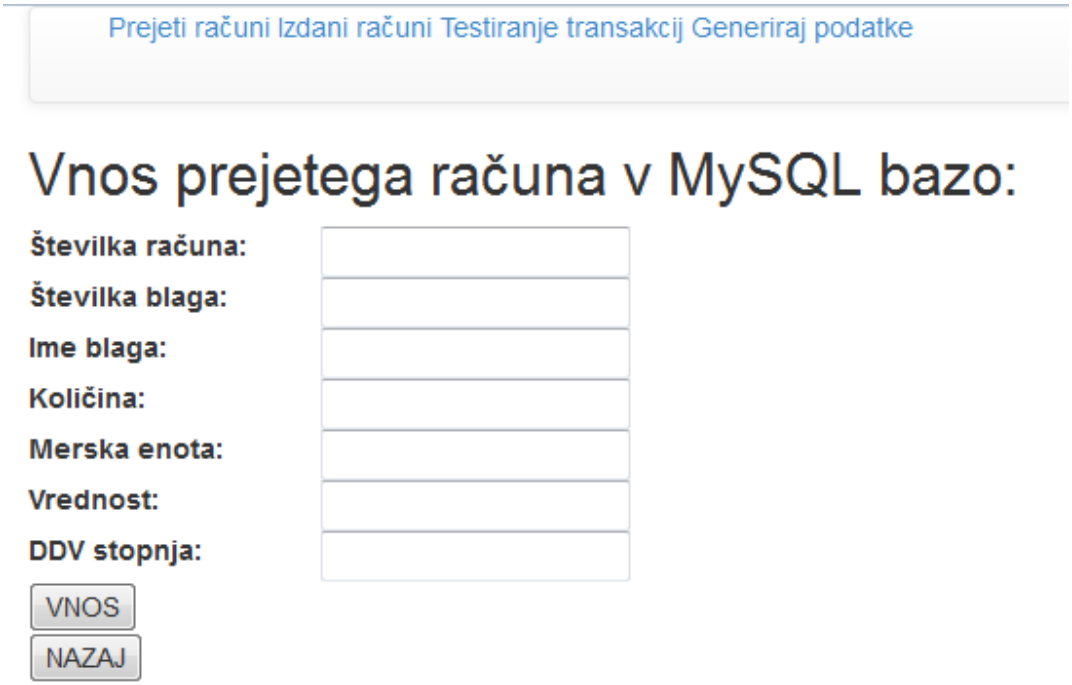
Vnos računa v MySQL bazo je razdeljen na tri korake. Vsi podatki iz računa morajo biti vneseni v pravilnem vrstnem redu. To je potrebno zaradi odvisnosti tabel v bazi. Slike 4.2, 4.3 in 4.4 prikazujejo te korake. S SQL stavkom INSERT INTO vnesemo podatke v MySQL bazo.



Slika 4.2: Vnos v MySQL bazo 1. korak.

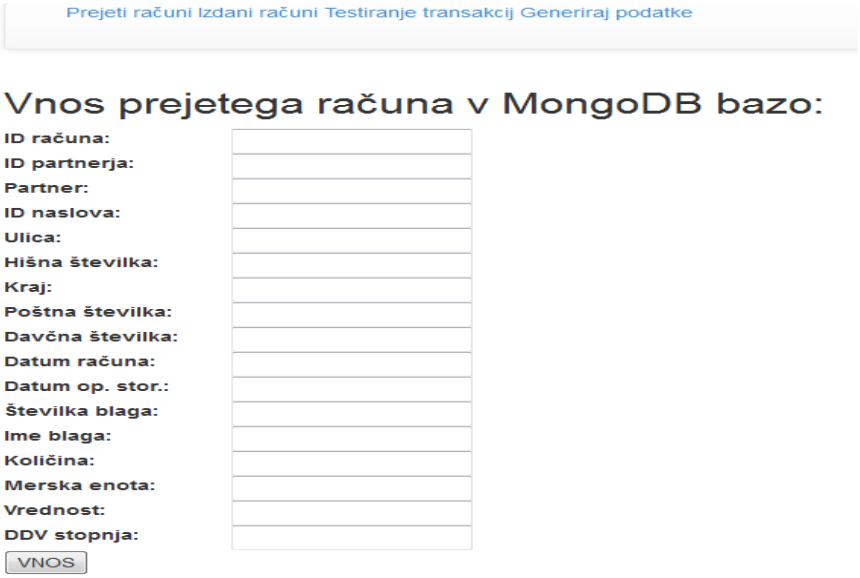


Slika 4.3: Vnos v MySQL bazo 2. korak.



Slika 4.4: Vnos v MySQL bazo 3. korak.

Pri vnašanju računa v bazo MongoDB ni potrebno vnašati v več korakih. Zbirke v bazi niso med seboj odvisne. V bazo vnesemo podatke z ukazom `insert()`.



Slika 4.5: Vnos računa v bazo MongoDB.

4.2 Testne poizvedbe za testiranje podatkovne baze MySQL

V tem poglavju bomo opisali poizvedbe, ki so napisane v SQL jeziku za povpraševanje po podatkih v podatkovni bazi MySQL. PHP programsko kodo, ki se ponavlja, bomo opisali samo prvič. Za potrebe testiranja smo ustvarili preprosto internetno aplikacijo preko katere smo pošiljali poizvedbe obema testiranima bazama. Za razvoj smo uporabili skriptni jezik PHP. Za skriptni jezik PHP smo se odločili, ker nudi zelo dobro podporo, tako za relacijsko bazo MySQL in dokumentno bazo MongoDB.

4.2.1 Poišči vse izdane račune za določenega partnerja (Epsilon d.o.o.)

SQL SELECT stavek izbere podatke iz baze MySQL. Izbere vse podatke iz tabele `izdani_racun`. Za izbiro uporabi tabeli `izdani_racun` in `partner`. To pa zato, ker tabela `izbrani_racun` ne vsebuje stolpca `naziv`, po katerem sprašujemo. Ta stolpec ima tabela `partner`. Povežemo obe tabeli s ključem in na koncu dodamo pogoj naziv partnerja.

```
$Sql = "SELECT izdani_racun.* FROM izdani_racun, partner
WHERE izdani_racun.ID_partnerja = partner.ID_partnerja AND
partner.naziv = 'Epsilon d.o.o.'";
```

SQL SELECT stavek php skripta posreduje MySQL bazi.

```
<?php
$dbc = mysqli_connect('localhost', 'dev', '**********',
'diploma')
OR die('Napaka:' . mysqli_connect_error());
```

Funkcija `mysqli_connect()` odpre novo povezavo do MySQL strežnika in shrani povezavo v spremenljivko `$dbc`. V funkcijo `mysqli_connect()` dodamo parametre:

- host (navedemo ime gostitelja ali njegov IP naslov),
- username (navedemo ime uporabnika baze MySQL),
- password (navedemo geslo do dostop baze MySQL),
- dbname (navedemo ime baze MySQL, do katere želimo dostopati)[13].

Funkcija `mysqli_connect_error()`, pa nas obvesti, o napaki, če nismo mogli odpreti nove povezave do MySQL strežnika[14].

```
$start = microtime(true);
```

Funkcija `microtime(true)` vrne število mikrosekund, ki pretekle od 1. Januarja 1970. Parameter v funkciji nastavimo na `true`, tako da dobimo namesto niza, število s plavajočo vejico. Število shranimo v spremenljivko `$start`.

```
$Sql = "SELECT izdani_racun.* FROM izdani_racun, partner
WHERE izdani_racun.ID_partnerja = partner.ID_partnerja
AND partner.naziv = 'Epsilon d.o.o.'";
$result = mysqli_query($dbc, $Sql);
```

Funkcija `mysqli_query()` izvede povpraševanje po bazi. Rezultat shranimo v spremenljivko `$result`. Funkcija zahteva dva parametra:

- connection (določi MySQL povezavo do baze),
- query (določi kateri SQL stavek se izvede)[16].

```
if($result === FALSE) {
    die(mysql_error());
}
```

Funkcija `mysql_error()` vrne opis napake, ki se je zgodila[15].

```
while ($row = mysqli_fetch_row($result)) {}
```

Funkcija `mysqli_fetch_row()` vrne vrstico, ki smo jo iskali. Funkcija zahteva en parameter:

- result (določimo vrednost, ki nam jo je vrnila funkcija `mysqli_query()`).

```
$end = microtime(true);
```

Funkcija `microtime(true)` vrne število mikrosekund, ki pretekle od 1. Januarja 1970. Parameter v funkciji nastavimo na `true`, tako da dobimo namesto niza, število s plavajočo vejico. Število shranimo v spremenljivko `$end`.

```
$total = $end - $start;
```

V spremenljivko `$total` shranimo vrednost, ki smo jih dobili z odštevanjem spremenljivk `$end` in `$start`. Dobili smo vrednost v mikrosekundah.

```
echo round(($total)*1000)." ms";
```

Funkcija `echo()` izpiše v brskalniku kar smo ji določili. Funkcija `round()` pa zaokroži podatek, ki je v obliki plavajoče vejice na celo število.

```
mysqli_close($dbc);
?>
```

Funkcija `mysqli_close()` prekine obstoječo povezavo do baze MySQL. Funkcija zahteva en parameter:

- connection (določi povezavo do baze MySQL, ki naj se prekine)[12].

4.2.2 Poišči vse prejete račune za določenega partnerja (Omega d.o.o.)

SQL SELECT stavek izbere podatke iz baze MySQL. Izbere vse podatke iz tabele `prejeti_racun`. Za izbiro uporabi tabeli `prejeti_racun` in `partner`. To pa zato, ker

tabela `prejeti_racun` ne vsebuje stolpca `naziv`, po katerem sprašujemo. Ta stolpec ima tabela `partner`. Povežemo obe tabeli s ključem in na koncu dodamo pogoj naziv partnerja.

```
$Sql = "SELECT prejeti_racun.* FROM prejeti_racun, partner
WHERE prejeti_racun.ID_partnerja = partner.ID_partnerja
AND partner.naziv = 'Omega d.o.o.'";
```

SQL SELECT stavek se pošlje preko PHP skripte bazi.

```
<?php
$dbc = mysqli_connect('localhost', 'dev', '**********',
'diploma')
OR die('Napaka:' . mysqli_connect_error());
$start = microtime(true);
$Sql = "SELECT prejeti_racun.* FROM prejeti_racun,
partner WHERE prejeti_racun.ID_partnerja =
partner.ID_partnerja AND partner.naziv = 'Omega d.o.o.'";
$result = mysqli_query($dbc, $Sql);
if($result === FALSE) {
    die(mysql_error());
}
while ($row = mysqli_fetch_row($result)) {}
$end = microtime(true);
$total = $end - $start;
echo round(($total)*1000)." ms";
mysqli_close($dbc);
?>
```

4.2.3 Kateri partnerji so kupili določeno blago (kava)

SQL SELECT stavek izbere podatke iz baze MySQL. Izbere podatke iz tabele `partner`. V tabeli `partner` pa stolpec `naziv`. Za izbiro uporabi tabele `blago`, `izdana_postavka`, `izdani_racun` in `partner`. To pa zato, ker tabela `partner` ne vsebuje stolpca `ime_blaga`, po katerem sprašujemo. Ta stolpec ima tabela `blago`. Povežemo vse tabele s ključi. Dodamo pogoj ime blaga je kava.

```
$Sql = "SELECT partner.naziv FROM blago, izdana_postavka,
izdani_racun, partner WHERE ime_blaga = 'kava' AND
blago.ID_blaga = izdana_postavka.ID_blaga AND
izdana_postavka.ID_iz_racuna = izdani_racun.ID_iz_racuna
AND izdani_racun.ID_partnerja = partner.ID_partnerja";
```

SQL SELECT stavek se pošlje preko PHP skripte bazi.

```
<?php
$dbc = mysqli_connect('localhost', 'dev', '**********',
'diploma')
OR die('Napaka:' . mysqli_connect_error());
$start = microtime(true);
$Sql = "SELECT partner.naziv FROM blago, izdana_postavka,
izdani_racun, partner WHERE ime_blaga = 'kava' AND
blago.ID_blaga = izdana_postavka.ID_blaga AND
izdana_postavka.ID_iz_racuna = izdani_racun.ID_iz_racuna
AND izdani_racun.ID_partnerja = partner.ID_partnerja";.
$result = mysqli_query($dbc, $Sql);
if($result === FALSE) {
    die(mysql_error());
}
while ($row = mysqli_fetch_row($result)) {}
$end = microtime(true);
$total = $end - $start;
echo round(($total)*1000)." ms";
mysqli_close($dbc);
?>
```

4.2.4 Kateri računi so bili izdani od - do (2014-8-7 in 2014-12-31)

SQL SELECT stavek izbere podatke iz baze MySQL. Izbere vse podatke iz tabele `izdani_racun`, ki ustrezajo pogoju, ki je podan. Za izbiro uporabi tabelo `izdani_racun`.

```
$Sql = "SELECT * FROM izdani_racun WHERE datum_iz_racuna
>= '2014-8-7' AND datum_iz_racuna <= '2014-12-31'";
```

SQL SELECT stavek se pošlje preko PHP skripte bazi.

```
<?php
$dbc = mysqli_connect('localhost', 'dev', '**********',
'diploma')
OR die('Napaka:' . mysqli_connect_error());
$start = microtime(true);
$Sql = "SELECT * FROM izdani_racun WHERE datum_iz_racuna
>= '2014-8-7' AND datum_iz_racuna <= '2014-12-31'";
$result = mysqli_query($dbc, $Sql);
if($result === FALSE) {
    die(mysql_error());
}
while ($row = mysqli_fetch_row($result)) {}
$end = microtime(true);
```

```
$total = $end - $start;
echo round(($total)*1000)." ms";
mysqli_close($dbc);
?>
```

4.2.5 Katero blago je bilo prodano za izbrani mesec (2014-8-1 in 2014-8-31)

SQL SELECT stavek izbere podatke iz baze MySQL. Izbere podatke iz tabele `blago`. V tabeli `blago` pa stolpec `ime_blaga`. Za izbiro uporabi tabele `blago`, `izdana_postavka`, `izdani_racun`. To pa zato, ker tabela `blago` ne vsebuje stolpca `datum_iz_racuna`, po katerem pogojujemo. Ta stolpec ima tabela `izdani_racun`. Povežemo vse tabele s ključi. Dodamo pogoj po katerem filtriramo iskanje.

```
$Sql = "SELECT blago.ime_blaga, FROM izdani_racun,blago,
izdana_postavka WHERE datum_iz_racuna >= '2014-8-1' AND
datum_iz_racuna <= '2014-8-31' AND
izdani_racun.ID_iz_racuna = izdana_postavka.ID_iz_racuna
AND izdana_postavka.ID_blaga = blago.ID_blaga";
```

SQL SELECT stavek se pošlje preko PHP skripte bazi.

```
<?php
$dbc = mysqli_connect('localhost', 'dev', '**********',
'diploma')
OR die('Napaka:' . mysqli_connect_error());
$start = microtime(true);
$Sql = "SELECT blago.ime_blaga, FROM izdani_racun,blago,
izdana_postavka WHERE datum_iz_racuna >= '2014-8-1' AND
datum_iz_racuna <= '2014-8-31' AND
izdani_racun.ID_iz_racuna = izdana_postavka.ID_iz_racuna
AND izdana_postavka.ID_blaga = blago.ID_blaga";
$result = mysqli_query($dbc, $Sql);
if($result === FALSE) {
    die(mysql_error());
}
while ($row = mysqli_fetch_row($result)) {}
$end = microtime(true);
$total = $end - $start;
echo round(($total)*1000)." ms";
mysqli_close($dbc);
?>
```

4.2.6 Poišči vse prejete račune

Ta SQL stavek je zelo enostaven. Izpišemo vse iz tabele `prejeti_racuni`.

```
$Sql = "SELECT * FROM prejeti_racun";
```

SQL SELECT stavek se pošlje preko PHP skripte bazi.

```
<?php
$dbc = mysqli_connect('localhost', 'dev', '**********',
'diploma')
OR die('Napaka:' . mysqli_connect_error());
$start = microtime(true);
$Sql = "SELECT * FROM prejeti_racun";
$result = mysqli_query($dbc, $Sql);
if($result === FALSE) {
    die(mysql_error());
}
while ($row = mysqli_fetch_row($result)) {}
$end = microtime(true);
$total = $end - $start;
echo round(($total)*1000)." ms";
mysqli_close($dbc);
?>
```

4.2.7 Poišči vse izdane račune

Ta SQL stavek je tudi zelo enostaven. Izpišemo vse iz tabele `izdani_racuni`.

```
$Sql = "SELECT * FROM izdani_racun";


<?php
$dbc = mysqli_connect('localhost', 'dev', '**********',
'diploma')
OR die('Napaka:' . mysqli_connect_error());
$start = microtime(true);
$Sql = "SELECT * FROM izdani_racun";
```

SQL SELECT stavek se pošlje preko PHP skripte bazi.

```
$result = mysqli_query($dbc, $Sql);
if($result === FALSE) {
    die(mysql_error());
}
while ($row = mysqli_fetch_row($result)) {}
```

```
$end = microtime(true);
$total = $end - $start;
echo round(($total)*1000)." ms";
mysqli_close($dbc);
?>
```

4.2.8 Poišči vso blago

SQL stavek izpiše vse iz tabele `blago`.

```
$Sql = "SELECT * FROM blago";
```

SQL SELECT stavek se pošlje preko PHP skripte bazi.

```
<?php
$dbc = mysqli_connect('localhost', 'dev', '**********',
'diploma')
OR die('Napaka:' . mysqli_connect_error());
$start = microtime(true);
$Sql = "SELECT * FROM blago";
$result = mysqli_query($dbc, $Sql);
if($result === FALSE) {
    die(mysql_error());
}
while ($row = mysqli_fetch_row($result)) {}
$end = microtime(true);
$total = $end - $start;
echo round(($total)*1000)." ms";
mysqli_close($dbc);
?>
```

4.3 Poizvedbe za testiranje baze MongoDB

Opisali bomo zahtevke, ki smo jih uporabili za testiranje baze MongoDB. Zahtevki so enaki, kot smo jih uporabili za testiranje baze MySQL. Baza MongoDB shranjuje dokumente v BSON formatu. BSON je sestavljenka iz besed binarno in JSON. JSON je okrajšava za JavaScript Object Notation. PHP programsko kodo, ki se ponavlja, bomo opisali samo prvič. Zaradi manjšega nabora funkcij, ki jih premore dokumentna baza, smo določene operacije izvajali v PHP skripti in ne v bazi sami.

4.3.1 Poišči vse izdane račune za določenega partnerja (Epsilon d.o.o.)

```
<?php
$m = new MongoClient();
```

Funkcija `new MongoClient()` omogoči povezavo do baze MongoDB.

```
$db = $m->selectDB('diploma');
```

`Funkcija selectDB()` določi katera podatkovna baza v MongoDB se bo uporabljala.

```
$partnerji = new MongoCollection($db, 'partnerji');
$prodaja = new MongoCollection($db, 'prodaja');
```

Funkcija `new MongoCollection()` določi katera zbirka se bo uporabljala.

```
$start = microtime(true);
$partnerQuery = array('naziv' => 'Epsilon d.o.o.');
$cursor = $partnerji->find($partnerQuery,
array('ID_partnerja' => 1, '_id' => 0));
```

Funkcija `find()` izvede zgornjo poizvedbo `$partnerQuery` in vrne stolpec `'ID_partnerja'`.

```
$partner = iterator_to_array($cursor);
```

`Funkcija iterator_to_array()` je PHP funkcija, ki spremeni rezultat poizvedbe v seznam.

```
$prodajaQuery = array('ID_partnerja' =>
(string) partner[0]['ID_partnerja']);
```

Sestavimo novo poizvedbo z uporabo rezultata prejšnje.

```
$cursor = $prodaja->distinct('ID_iz_racuna',
$prodajaQuery);
```

V seznam `$cursor` vrne vse vrstice iz zbirke `prodaja` z unikatno vrednostjo iz stolpca `'ID_iz_racuna'`, katerih vrednost `'ID_partnerja'` se nahaja v zgornji poizvedbi.

```
$end = microtime(true);
$total = $end - $start;
echo "<p style='color:red;'>".round(($total)*1000)." ms";
?>
```

4.3.2 Poišči vse prejete račune za določenega partnerja (Omega d.o.o.)

Izvede se poizvedba, ki išče naziv, ki ima vrednost Omega d.o.o. in vrne stolpec `'ID_partnerja'`.

```
$cursor=$partnerji->find($partnerQuery,
array('ID_partnerja' => 1, '_id' => 0));
```

V seznam `$cursor` vrne vse vrstice iz zbirke `'nakupi'` z unikatno vrednostjo iz stolpca `'ID_pr_racuna'`, katerih vrednost `'ID_partnerja'` se nahaja v zgornji poizvedbi.

```
$cursor = $nakupi->distinct('ID_pr_racuna', $nakupQuery);
```

```
<?php
$m = new MongoClient();
$db = $m->selectDB('diploma');
$partnerji = new MongoCollection($db, 'partnerji');
$nakupi = new MongoCollection($db, 'nakupi');
$start = microtime(true);
$partnerQuery = array('naziv' => 'Omega d.o.o.');
$cursor = $partnerji->find($partnerQuery,
array('ID_partnerja' => 1, '_id' => 0));
$partner = iterator_to_array($cursor);
$nakupQuery = array('ID_partnerja' => (string)
$partner[0]['ID_partnerja']);
```

PHP skripta sestavi novo poizvedbo z uporabo rezultata prejšnje.

```
$cursor = $nakupi->distinct('ID_pr_racuna', $nakupQuery);
$end = microtime(true);
$total = $end - $start;
echo "<p style='color:red;'>".round(($total)*1000)." ms";
?>
```

4.3.3 Kateri partnerji so kupili določeno blago (kava)

Izvede se poizvedba `$blagoQuery` in vrne stolpec `'ID_partnerja'`.

```
$cursor = $prodaja->find($blagoQuery, array('ID_partnerja'
=> 1, '_id' => 0));
```

Sestavi novo poizvedbo z uporabo seznama vrednosti `'ID_partnerja'`.

```
$partnerQuery = array('ID_partnerja' => array('$in' =>
$partnerIDArray));
```

Izvede zgornjo poizvedbo in vrne stolpec `'naziv'`.

```
$cursor = $partnerji->find($partnerQuery, array('naziv' =>
1, '_id' => 0));
```

```
<?php
$m = new MongoClient();
$db = $m->selectDB('diploma');
$partnerji = new MongoCollection($db, 'partnerji');
$prodaja = new MongoCollection($db, 'prodaja');
$start = microtime(true);
$blagoQuery = array('ime_blaga' => 'kava');
$cursor = $prodaja->find($blagoQuery, array('ID_partnerja'
=> 1, '_id' => 0));
$partnerArray = iterator_to_array($cursor);
$partnerIDArray = array();
```

Funkcija `array()` je PHP funkcija in ustvari nov prazen seznam.

```
foreach ($partnerArray as $key => $value) {
```

Sprehod po seznamu `$partnerArray`.

```
if (!array_search($value['ID_partnerja'],
$partnerIDArray)) {
```

Pogoj, ki preverja, če seznam `$partnerIDArray` že vsebuje vrstico `$value['ID_partnerja']`.

```
$partnerIDArray[] = $value['ID_partnerja'];
```

Če vrstice ni, jo doda.

```
}}$partnerQuery = array('ID_partnerja' => array('$in' =>
$partnerIDArray));

$cursor = $partnerji->find($partnerQuery, array('naziv' =>
1, '_id' => 0));

foreach ($cursor as $row) {}

$end = microtime(true);

$total = $end - $start;
echo "<p style='color:red;'>".round(($total)*1000)." ms";
?>
```

4.3.4 Kateri računi so bili izdani od - do (2014-8-7 in 2014-12-31)

Sestavimo poizvedbo, ki v zbirki poišče vse dokumente, ki imajo `'datum_iz_racuna'` med 2014-08-07 in 2014-12-31.

```
$prodajaQuery = array("datum_iz_racuna" =>array('$gte' =>
"2014-08-07",'$lte' => '2014-12-31'));
```

Vrnemo seznam vrednosti iz zbirke `prodaja`, kjer vrednosti izpolnjujejo pogoj, ki je podan.

```
$cursor=$prodaja->distinct('ID_iz_racuna', $prodajaQuery);

<?php
$m = new MongoClient();
$db = $m->selectDB('diploma');
$prodaja = new MongoCollection($db, 'prodaja');
$start = microtime(true);
$prodajaQuery = array("datum_iz_racuna" =>array('$gte' =>
"2014-08-07",'$lte' => '2014-12-31'));
$cursor = $prodaja->distinct('ID_iz_racuna',
$prodajaQuery);
$end = microtime(true);
$total = $end - $start;
echo "<p style='color:red;'>".round(($total)*1000)." ms";
?>
```

4.3.5 Katero blago je bilo prodano za izbrani mesec (2014-8-1 in 2014-8-31)

Sestavimo poizvedbo, ki v zbirki poišče vse dokumente, ki imajo `'datum_iz_racuna'` med 2014-08-01 in 2014-08-31.

```
$prodajaQuery = array("datum_iz_racuna" =>array('$gte' =>
"2014-08-01",'$lte' => '2014-08-31'));
```

Izvede zgornjo poizvedbo in vrne stolpec `'ime_blaga'`.

```
$cursor = $prodaja->find($prodajaQuery, array('ime_blaga'
=> 1, '_id' => 0));
```

```
<?php
$m = new MongoClient();
$db = $m->selectDB('diploma');
$prodaja = new MongoCollection($db, 'prodaja');
$start = microtime(true);
$prodajaQuery = array("datum_iz_racuna" =>array('$gte' =>
"2014-08-01",'$lte' => '2014-08-31'));
$cursor = $prodaja->find($prodajaQuery, array('ime_blaga'
=> 1, '_id' => 0));
foreach ($cursor as $row) {};
$end = microtime(true);
$total = $end - $start;
echo "<p style='color:red;'>".round(($total)*1000)." ms";
?>
```

4.3.6 Poišči vse prejete račune

V seznam `$cursor` vrne vse vrstice iz zbirke `nakupi` z unikatno vrednostjo iz stolpca `'ID_pr_racuna'`.

```
$cursor = $nakupi->distinct('ID_pr_racuna');.
```

```
<?php
$m = new MongoClient();
$db = $m->selectDB('diploma');
$nakupi = new MongoCollection($db, 'nakupi');
$start = microtime(true);
$cursor = $nakupi->distinct('ID_pr_racuna');
$end = microtime(true);
$total = $end - $start;
echo "<p style='color:red;'>".round(($total)*1000)." ms";
```

```
?>
```

4.3.7 Poišči vse izdane račune

V seznam `$cursor` vrne vse vrstice iz zbirke prodaja z unikatno vrednostjo iz stolpca `'ID_iz_racuna'`.

```
$cursor = $prodaja->distinct('ID_iz_racuna');
```

```
<?php
$m = new MongoClient();
$db = $m->selectDB('diploma');
$prodaja = new MongoCollection($db, 'prodaja');
$start = microtime(true);
$cursor = $prodaja->distinct('ID_iz_racuna');
$end = microtime(true);
$total = $end - $start;
echo "<p style='color:red;'>".round(($total)*1000)." ms";
?>
```

4.3.8 Poišči vso blago

V seznam `$seznamBlaga` vrne vse vrstice iz zbirke `prodaja` z unikatno vrednostjo iz stolpca `'ID_blaga'`.

```
$seznamBlaga = $prodaja->distinct('ID_blaga');
```

```
<?php
$m = new MongoClient();
$db = $m->selectDB('diploma');
$prodaja = new MongoCollection($db, 'prodaja');
$start = microtime(true);
$seznamBlaga = $prodaja->distinct('ID_blaga');
$end = microtime(true);
$total = $end - $start;
echo "<p style='color:red;'>".round(($total)*1000)." ms";
?>
```

5. Predstavitev in razlaga rezultatov testiranja

5.1 Predstavitev rezultatov testiranja

Bazi smo testirali s pomočjo osmih poizvedb. Poizvedbe so take, da potrebujejo podatke iz ene ali več tabel iz baze MySQL, oziroma zbirk iz baze MongoDB. Poizvedbe smo petkrat ponovili, najhitrejši in najpočasnejši čas smo izločili. Iz ostalih treh časov, pa smo izračunali povprečje.

Velikost tabel v MySQL bazi je predstavljena v spodnji Tabeli 5.1.

Ime tabele	**Število vrstic v tabeli**
blago	110
izdana_postavka	120000
izdani_racun	60000
naslov	7
partner	24
posta	9
prejeta_postavka	120000
prejeti_racun	60000

Tabela 5.1: Število vrstic v posameznih tabelah.

V tabeli 5.2 so predstavljeni podatki o številu dokumentov v posamezni zbirki v bazi MongoDB.

Ime zbirke	**Število dokumentov v zbirki**
nakupi	120032
partnerji	24
prodaja	119969

Tabela 5.2: Število dokumentov v posameznih zbirkah.

V tabeli 5.3 smo zbrali podatke testiranja baze MongoDB. Čas izvajanja je merjen v milisekundah.

Poizvedba	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Test 5**	**Povprečje**
Izpiši vse izdane račune za določenega partnerja (Epsilon d.o.o.).	256	~~264~~	~~253~~	261	263	**260**
Izpiši vse prejete račune za določenega partnerja (Omega d.o.o.).	258	~~270~~	253	~~250~~	252	**254**
Kateri partnerji so kupili določeno blago (kava).	~~151~~	162	~~163~~	157	153	**157**
Kateri računi so bili izdani od - do (2014-8-7 in 2014-12-31).	632	~~644~~	637	627	~~627~~	**632**
Izpiši vso prodano blago	~~362~~	~~379~~	376	367	368	**370**

za izbrani mesec (2014-8-1 in 2014-8-31).						
Izpiši vse prejete račune.	~~632~~	653	~~674~~	643	662	**653**
Izpiši vse izdane račune.	~~697~~	729	~~762~~	713	719	**720**
Izpiši vse blago.	231	226	~~238~~	222	~~219~~	**226**

Tabela 5.3: Rezultati testiranja baze MongoDB.

V tabeli 5.4 smo zbrali podatke testiranja baze MySQL. Čas izvajanja je merjen v milisekundah.

Poizvedbe	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Test 5**	**Povprečje**
Izpiši vse izdane račune za določenega partnerja (Epsilon d.o.o.).	~~35~~	25	23	22	~~22~~	**23**
Izpiši vse prejete račune za določenega partnerja (Omega d.o.o.).	~~45~~	32	~~30~~	39	37	**36**
Kateri partnerji so kupili določeno blago (kava).	~~51~~	49	48	49	~~48~~	**49**
Kateri računi so bili izdani od - do (2014-8-7 in 2014-12-31).	~~121~~	124	122	~~132~~	124	**123**
Izpiši vso prodano blago za izbrani mesec (2014-8-1 in 2014-8-31).	~~258~~	304	306	301	~~314~~	**304**

Izpiši vse prejete račune.	~~323~~	~~311~~	315	315	312	**314**
Izpiši vse izdane račune.	~~333~~	~~332~~	332	333	332	**332**
Izpiši vse blago.	~~32~~	~~31~~	32	31	32	**32**

Tabela 5.4: Rezultati testiranja baze MySQL.

5.2 Razlaga rezultatov testiranja in sklepne ugotovitve

V tabeli 5.5 podajamo povprečne čase v milisekundah za obe testirani bazi MongoDB in MySQL.

Poizvedba	**Povprečni čas MongoDB**	**Povprečni čas MySQL**
1. Izpiši vse izdane račune za določenega partnerja (Epsilon d.o.o.).	**260**	**23**
2. Izpiši vse prejete račune za določenega partnerja (Omega d.o.o.).	**254**	**36**
3. Kateri partnerji so kupili določeno blago (kava).	**157**	**49**
4. Kateri računi so bili izdani od - do (2014-8-7 in 2014-12-31).	**632**	**123**
5. Izpiši vso prodano blago za izbrani mesec (2014-8-1 in 2014-8-	**370**	**304**

31).		
6. Izpiši vse prejete račune.	**653**	**314**
7. Izpiši vse izdane račune.	**720**	**332**
8. Izpiši vse blago.	**226**	**32**

Tabela 5.5: Povprečni časi poizvedb testiranih baz.

Vsako testirano poizvedbo je baza MySQL izvršila hitreje kot baza MongoDB. Vendar kljub temu baza MongoDB ni počasnejša. Vedeti moramo, da imajo posamezne tabele v bazi MySQL manjše število vrstic, kot pa imajo zbirke dokumentov.

Poizvedba 1, v bazi MySQL išče po dveh tabelah ki imata skupaj 60000 vrstic baza MongoDB pa išče po dveh zbirkah, ki imata 120000 dokumentov. Kljub temu, da mora baza MongoDB preiskati še enkrat več dokumentov, pa je počasnejša za desetkrat.

Pri poizvedbi 2 imamo podobno situacijo. Baza MongoDB je počasnejša približno za desetkrat.

Pri poizvedbi 3 pa baza MySQL išče po treh tabelah ki imajo skupaj približno 180000 vrstic. Baza MongoDB pa išče po dveh zbirkah, ki imata skupaj približno 120000 dokumentov. Tu pa razlika ni več tako velika. Kljub temu da je več podatkov v bazi MySQL je baza MySQL trikrat hitrejša od baze MongoDB.

Poizvedbo 4 je baza MySQL izvršila približno petkrat hitreje. Obe bazi iščeta samo iz ene tabele oziroma zbirke. Zbirka je zopet še enkrat večja od tabele. Pri tej poizvedbi je potrebno pri bazi MongoDB odstraniti podvojene podatke, kar dodatno upočasni bazo.

Baza MySQL pri poizvedbi 5 išče iz treh tabel. Baza MongoDB pa iz ene zbirke. Poizvedbi sta skoraj enako hitri. V tem primeru mora baza MySQL ustvariti več stikov med tabelami, tako, da jo to upočasni. Na drugi strani pa baza MongoDB nima potrebe po stikih, ker so vsi potrebni podatki že v zbirki. Problemov s podvojenimi podatki pa v tem primeru ni.

Čeprav je v poizvedbi 6 nominalno hitrost baze MySQL enkrat hitrejša od baze MongoDB, pa išče baza MySQL po 60000 vrsticah, baza MongoDB pa po 120000 dokumentih. V tem primeru mora baza MongoDB še dodatno izločati podvojene podatke.

Poizvedba 7 je vrnila podobne rezultate kot poizvedba 6.

Pri poizvedbi 8 pa je baza MongoDB sedemkrat počasnejša. Vendar v tem primeru baza MySQL išče po tabeli velikosti 110 vrstic. Baza MongoDB pa išče po zbirki velikosti 120000 dokumentov in potrebno je še izločati podvojene podatke.

6. Zaključek

V okviru testiranja smo ugotovili, da je za potrebe računovodske aplikacije, za upravljanje z manjšimi količinami podatkov, veliko bolj primerna relacijska podatkovna baza MySQL. Podatkovna baza MongoDB pa deluje pri enostavnem iskanju med ogromno količino podatkov veliko bolje, kar je pokazala poizvedba 8. Podjetja, ki si želijo v svoje delovanje čim prej vnesti nove programske rešitve (ang. early adopters) želijo hitro implementacijo novega produkta. Da bi to dosegli, pogosto preslikajo obstoječo shemo relacijske podatkovne baze neposredno v dokumentno bazo. To pa ni optimalno. Zaradi manjšega nabora funkcij, ki jih ima dokumentna baza, moramo operacije izvajati v aplikaciji, kar podaljša čas same transakcije. Za optimizacijo NoSQL podatkovne baze pa je potrebno veliko časa in znanja. Predvidevamo, da ni veliko podjetij, ki bi morali hraniti na milijone prejetih in izdanih računov, vsaj ne v aktivni bazi. NoSQL nima polne podpore ACID, kar pa je pri računovodskih transakcijah nujno potrebno. Zato zaključujemo, da podjetjem ni potrebno menjavati obstoječih rešitev, ki temeljijo na bazah SQL. V večini primerov se podjetjem ne izplača investicija, ker dobljeni rezultati ne bodo toliko večji, če sploh.

Podatkovna baza MongoDB je iz verzije v verzijo boljša. Razvijalci dodajajo vedno nove in nove funkcionalnosti. Predvidevamo, da bo baza v prihodnosti postala zanimiva tudi za računovodske aplikacije. Glavna zahteva, da to postane, je polna podpora principu ACID.

Literatura

Viri:

[1] Beaulieu Alan, Learning SQL, second edition, Sebastopol, 2009.

[2] Chodorow Kristina and Dirolf Michael, MongoDB: The Definitive Guide, Sebastopol, 2010.

Internetni viri:

[3] Database models. (avgust 2014). [Online]. Dosegljivo: http://www.unixspace.com/context/databases.html.

[4] Document-oriented database. (avgust 2014). Wikipedia. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Document-oriented_database.

[5] E–gradiva. (avgust 2014). [Online]. Dosegljivo: http://www.egradiva.net/moduli/podatkovne_baze/03_definicija/01_datoteka.html.

[6] Hierarchical database model. (september 2014). [Online]. Dosegljivo: http://en.wikipedia.org/wiki/Hierarchical_database_model.

[7] JSON. (avgust 2014). Wikipedia. [Online]. Dosegljivo: http://en.wikipedia.org/wiki/JSON.

[8] MongoDB. (avgust 2014). Wikipedia. [Online]. Dosegljivo: http://www.en.wikipedia.org/wiki/MongoDB.

[9] MongoDB. (avgust 2014). [Online]. Dosegljivo: http://www.mongodb.org/downloads.

[10] MySQL. (september 2014). Wikipedija. [Online]. Dosegljivo: http://sl.wikipedia.org/wiki/MySQL.

[11] Network model. (september 2014). [Online]. Dosegljivo: http://en.wikipedia.org/wiki/Network_model.

[12] PHP mysqli_close() Function. (september 2014). W3schools.com. [Online]. Dosegljivo: http://www.w3schools.com/php/func_mysqli_close.asp.

[13] PHP mysqli_connect() Function. (september 2014). W3schools.com. [Online]. Dosegljivo: http://www.w3schools.com/php/func_mysqli_connect.asp.

[14] PHP mysqli_connect_error() Function. (september 2014). W3schools.com. [Online]. Dosegljivo: http://www.w3schools.com/php/func_mysqli_connect_error.asp.

[15] PHP mysqli_error() Function. (september 2014). W3schools.com. [Online]. Dosegljivo: http://www.w3schools.com/php/func_mysqli_error.asp.

[16] PHP mysqli_query() Function. (september 2014). W3schools.com. [Online]. Dosegljivo: http://www.w3schools.com/php/func_mysqli_query.asp.

[17] SLO-POI.si. (september 2014). [Online]. Dosegljivo: http://www.slo-poi.si/projekt.php.

[18] Spletni strežnik Apache. (avgust 2014). Wikipedija. [Online]. Dose4gljivo: http://sl.wikipedia.org/wiki/Spletni_stre%C5%BEnik_Apache.

[19] SQL. (september 2014). Wikipedija. [Online]. Dosegljivo: http://sl.wikipedia.org/wiki/SQL.

[20] Sybase®. (avgust 2014). SAP Sybase PowerDesign [Online]. Dosegljivo: http://www.sybase.com/products/modelingdevelopment/powerdesigner.